Sources of Evidence for Retrieval of Scientific Articles

Using term frequency methods or Tf-Idf are the most commonly used sources of evidence for IR systems. Although very effective for general purposes, it is often required to tailor global or local evidences according to domain or use-case, in order to obtain better results. In modern IR models, it is expected that the system should be able to fetch results with greater accuracy and with the minimal amount of efforts. Hence, it becomes important to understand the task and process the incoming query as per the the information need.

One such domain is the retrieval of scientific articles from internet or any database in general. A very common drawback of directly using tf-idf for this task can be that since articles are considerably long, a lot many papers belonging to the same domain can have the same set of terms, which will require more effort or specific terms to distinguish between them. If a user tries to look for a paper of any particular author, there can be many papers by the same author, in collaboration with many other authors from same and different domains, resulting in a large retrieved set and low precision. What makes it worse is that many users mistype the author names, or more common is that authors can have varying names across literature (full names or initials or combination of both), making it difficult to retrieve relevant documents. For these scenarios, we will require some way to improve the performance of the system. Following are two such sources of evidence which attempt to resolve the above presented problems:

(i) Term Count of words present in the Title of an article: While searching for any articles, users search for relevant articles and papers using keywords or topics that are defined in the Title of the article. For e.g. Users searching for "Sentiment Analysis" articles. The title is a single line description for the whole article and if the query terms match with the title, it is highly probable to be relevant to user's query. The weighting scheme that can be used to incorporate term count of query Q with the title of the article document D can be defined as,

$$\boldsymbol{tc(Q, D) = Count[\ Q \cap Title_D\ ]} \qquad \textbf{... eq.(i)}$$

where,

$\boldsymbol{Q, D}$ = query input and document respectively
$\boldsymbol{Title_D}$ = Title of document D
$\boldsymbol{Count[\ Q \cap Title_D\ ]}$ = Count of number of terms common in query and document title.

For getting a better idea of relevance with respect to title, stopwords from the title and query can be removed. Also, another important point to note is that usually the length of the titles of scientific articles in general doesn't exceed 12-15 terms, which is also the approximate range of maximum value for tc(Q, D). eq.(i) can be normalized using the log function to keep the weights in lower ranges as follows:

$$\boldsymbol{tc(Q, D) = log_{10}\ (Count[\ Q \cap Title_D\ ])} \qquad \textbf{... eq.(ii)}$$

(ii) Presence of author names in the query: It is also very common for users to look for articles using the author's names, or to look for articles written by some author(s). This feature can be used to identify relevance between query and documents. A simple weighting scheme considering just the presence of author name in query can be defined as a score:

$$\boldsymbol{S(Q, D) = \sum_{t \in T_Q} f(t, D)} \qquad \textbf{... eq.(iii)}$$

where,

$\boldsymbol{Q, D}$ = Query and Document respectively
$\boldsymbol{T_Q}$ = Set of terms of query Q

$$f(t, D) = \begin{cases} 1, & t \text{ is present in authors of } D \\ 0, & Otherwise \end{cases}$$

The above count can be normalized by dividing f(t, D) with the overall number of authors present in document D. But since the log function has been used for normalizing tc(Q, D), log can be used. The equation then becomes as follows:

$$S(Q, D) = log_{10}\left(\sum_{t \in T_Q} f(t, D)\right) \quad \text{... eq.(iv)}$$

The above weighting scheme for author considers a boolean answer, i.e. query term present as an author or not. However, it can be prone to wrong score for the following scenarios:

- It is common for users to mistype the name of any author or type it partially with initials.
- Same authors use initials of their first or middle name or write their name differently as well across different scientific articles.

These above issues can also be solved by introducing a string matching based weighting scheme. Using Levenshtein distance as a base for string matching, we can define our own version of normalized distance between two terms as:

$$dist(t_1, t_2) = \frac{0.1 + LD(t_1, t_2)}{len_{t_1} + len_{t_2}} \quad \text{... eq.(v)}$$

where,

$LD(t_1, t_2)$ = Levenshtein distance between terms t1 and t2
len_{t_1}, len_{t_2} = length of terms t1 and t2 respectively

A constant of 0.1 is added to consider for the exact matching terms (where author name is present in the query). Using eq.(v) as the measure, distance matrix can be obtained between all query terms and authors of a document. Since our focus is to look at the minimum distance (i.e. closest matching term in the query to any of the authors), only the the minimum of this matrix can be considered. So far in this method, all the documents having same minimum value d in distance matrix carry the same weight. This can be further refined by multiplying this by the fraction of what can be called as the relative ratio of authors for a document D which can be defined as:

$$Relative\ Ratio, R(D) = \frac{n_D}{N} \quad \text{... eq.(vi)}$$

where,

n_D = Number of authors for document D
N = Maximum number of authors of a document in our corpus

Multiplying the minimum distance d with the relative ratio R will have an effect of prioritizing the documents having less number of authors. For instance, if the user mentions an author name "ABC", and author "ABC" has three articles in corpus where for one he is the only author, one with another author "XYZ" and the third one with "DEF" and "PQRS", then using the distance measure from eq.(v), the minimum value from distance matrix will be 0.1/6 in all three cases (since query term "ABC" matches exactly to the author in these articles). But if we multiply relative ratios of these documents to the returned minimum distance 0.1/6, the value for the articles will be as follows:

Author{ABC} < Author{ABC, XYZ} < Author{ABC, DEF, PQRS}

For assigning highest weight to the Author{ABC}, we can subtract these values obtained from 1 as all values lie between 0 and 1. Hence, the weighting scheme W with the inclusion of author information considering the string matching as well becomes as follows:

$$W(Q, D) = 1 - (min[\, dist(t_1, t_2)\ \forall\ t_1 \in Q, \forall\ t_2 \in A_D]) \times R(D) \quad \text{... eq.(vii)}$$

where,

$\boldsymbol{A_D}$ = Author terms for the document D
$\boldsymbol{min}[\ ]$ = minimum of all the values inside the square brackets

Substituting the values, W becomes,

$$\boldsymbol{W(Q, D) = 1 - (min[\ \frac{0.1 + LD(t_1, t_2)}{len_{t_1} + len_{t_2}}\ \forall\ t_1 \in Q, \forall\ t_2 \in A_D])} \times \frac{n_D}{N} \quad \textbf{... eq.(viii)}$$

The weight W(Q, D) obtained in eq.(viii) and the normalized tc(Q, D) obtained in eq.(i) can be added to give the document score for these features.